UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/760,031	01/12/2001	Robert H. Halstead JR.	2682.1014-004	1846

21005 7590 01/22/2004

HAMILTON, BROOK, SMITH & REYNOLDS, P.C.
530 VIRGINIA ROAD
P.O. BOX 9133
CONCORD, MA 01742-9133

EXAMINER	
KANG, INSUN	
ART UNIT	PAPER NUMBER
2124	7

DATE MAILED: 01/22/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/760,031	HALSTEAD ET AL.	
	Examiner	Art Unit	
	Insun Kang	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *1/12/20, 1/4/2002, 2/12/2002, 1/27/2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *12 January 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☒ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

13)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) *2,3,5,6*.
4) ☐ Interview Summary (PTO-413) Paper No(s). _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. This action is responding to application papers dated 1/12/20, 1/4/2002, 2/12/2002, 1/27/2003, and 9/10/2003.

2. Claims 1-25 are pending.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

4. Claims 1-25 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting over claims 1-10, 13-22 and 25-28 copending Application No.09/759697.

Although the conflicting claims are not identical, they are not patentably distinct from each other because they are directed to substantially the same invention and recites only obvious differences which would have been obvious to one of ordinary skill in the art of program development at the time of invention such as simply (i) omitting/adding steps or elements along with their functions, and/or (ii) implementing

the method steps with means for performing the steps, and/or (iii) computer program implementation of the method, and/or (iv) implementing a system, product and data signal having computer program for performing the method steps, as explained below.

The corresponding claims are as follows:

Instant claim:	copending claims:
1	1, 10, 13,22, 25,26,28
2	2, 14
3	3, 15
4	1, 13
5	4, 16
6	5, 17
7	6, 18
8	7, 19, 27
9	8, 20
10	9, 21
11	1,10, 13,22, 25,26,28
12	14, 2
13	15, 3
14	13, 1
15	16, 4
16	17, 5
17	18, 6
18	19, 7
19	20, 8
20	21, 9
21	1,10, 13,22, 25,26, 28
22	1,10, 13,22, 25,26, 28
23	6, 18
24	27, 19, 7
25	1,10, 13,22, 25,26, 28

Per claim 1:

Copending claim 1 recites **a method of processing data comprising: defining an object with defined with an option data structure which supports references to**

option values without preallocation of memory space for the full option values ("A method of processing data comprising: defining a class which supports an option data structure having, in instances of the class, references to option values without preallocation of memory space for the full option values"), as recited in instant claim 1. The copending claim 1 does not explicitly recite the additional limitations, as further recited in instant claim 1:

i) **defining an object with defined fields to support values in preallocated memory space** as recited in instant claim 1, but Official Notice is taken that this step was well known in the art of object oriented program development at the time of invention was made: A class definition typically includes one or more fields that declare and store data of various types. The compiler assigns a memory location (preallocate) to each field in the program based on its type accordingly and then the attribute value of the field is stored in the memory space preallocated to that field -- each instance of a class (object) has its own copy of the fields defined in the class. Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the step recited in the co-pending claim by adding the step of defining object with defined fields for the purpose of defining and storing data in the memory space preallocated by the compiler. Alternatively, the omission of defined fields from the instant method would be obvious because option values for the purpose of expediting the method.

Ii) **accessing a field value and accessing an option value in the object using expressions of the same syntactic form**, as further recited in instant claim 1. The co-

pending claim 1 does not explicitly recite this step. However, both field values and option values are property values of an instance object or a class. Official Notice is taken that this definition was well known in the art of object oriented program development at the time of invention was made: in object oriented programming, properties are usually made as private so that only member methods have access to the properties for purpose of encapsulation. Accessors (getters/setters) are typical of a well-designed object used for accessing a property of a class/object. Regardless of the different memory allocation methods of option values and field values, the use of the same syntax would be obvious because only member methods can access both option and field values. For example, the member method call syntax is objectName.methodName() in Java or C++. The dot operator is used between class/object name and method name: objectName.getValue(); objectName.setValue(field, option); objectName.fieldValue; objectName.optionValue. It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the step recited in the co-pending claim 1 by adding the step of **accessing a field value and accessing an option value in the object using expressions of the same syntactic form,** as recited in instant claim 1 because both option and field values are properties of the object.

The instant claim does not explicitly recite the step of **during compilation, using the type description in the option data structure to process an operation on the option value,** as recited in co-pending claim 1. It would have been obvious for one of ordinary skill in the art of program development at the time the instant

invention was made to modify the co-pending method by omitting the step of **during compilation, using the type description in the option data structure to process an operation on the option value** recited in co-pending claim 1 for the purpose of expediting the method.

Per claim 2:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding change handler code which corresponds to co-pending claim 2.

Per claim 3:

The rejection of claim 2 is incorporated, and further, the instant claim recites the additional limitation regarding change handler code for one option defined in different classes within a class inheritance hierarchy and from each class executed when the option value changes which corresponds to copending claim 3.

Per claim 4:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding the type description in the option data structure to process an operation on the option value which corresponds to copending claim 1.

Per claim 5:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding getting the set option value and getting the default value for the class which corresponds to copending claim 4.

Per claim 6:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding defining a first class and second class and encoding an option operation which corresponds to copending claim 5.

Per claim 7:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding notifying objects of a change in an option value and the option binding which corresponds to copending claim 6.

Per claim 8:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding a linked list of option items which corresponds to copending claim 7.

Per claim 9:

The rejection of claim 1 is incorporated, and further, the instant claim recites the additional limitation regarding nonlocal option hierarchy which corresponds to copending claim 8.

Per claim 10:

The rejection of claim 9 is incorporated, and further, the instant claim recites the additional limitation regarding graphical hierarchy which corresponds to copending claim 9.

Per claims 11-21:

The rejection of claims 1-10 is respectively incorporated, and further, the instant claims recite a system corresponding to copending claims 13-18, 20-22 and 25 respectively, modified in the manner set forth above in connection with claims 1-10 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth above with a system including means for performing the steps of the copending method.

Per claim 22:

The rejection of claims 1-21 is respectively incorporated, and further, the instant claim recites a program product corresponding to copending claim 26, respectively, modified in the manner set forth above in connection with claims 1-21 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth

above with a program product including means for performing the steps of the copending method.

Per claim 23:

The rejection of claims 1, 7, 11, 17 and 22 is respectively incorporated, and further, the instant claim recites the additional limitation regarding instructions to notify objects of a change in an option value which corresponds to the step of copending claims 6 and 18, except that the instant claim does not explicitly recite how to notify objects. It would have been obvious for one of ordinary skill in the art of program development at the time the instant invention was made to modify the copending method by omitting the method of notifying objects in copending claims 6 and 18 for the purpose of expediting the method.

Per claim 24:

The rejection of claims 1, 8 and 18 is respectively incorporated, and further, the instant claims recite a program product corresponding to copending claim 27, respectively, modified in the manner set forth above in connection with claims 1, 8 and 18 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth above with a program product including means for performing the steps of the copending method.

Per claim 25:

The rejection of claims 1, 11, 21 and 22 is respectively incorporated, and further, the instant claims recite a data signal corresponding to copending claims 28 respectively, modified in the manner set forth above in connection with claims 1, 11, 21 and 22 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth above with a data signal including means for performing the steps of the copending method.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

5. Claims 7, 17 and 23 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting over claims 1, 10, 19, 20 and 22 of copending Application No. 09/759695.

Although the conflicting claims are not identical, they are not patentably distinct from each other because they are directed to substantially the same invention and recites only obvious differences which would have been obvious to one of ordinary skill in the art of program development at the time of invention such as simply (i) omitting/adding steps or elements along with their functions, and/or (ii) implementing the method steps with means for performing the steps, and/or (iii) computer program implementation of the method, and/or (iv) implementing a system and product having computer program for performing the method steps, as explained below.

The corresponding claims are as follows:

Instant claim:	copending claims:
7	1, 10, 19, 20 and 22
17	1, 10, 19, 20 and 22
23	1, 10, 19, 20 and 22

Per claim 7:

Copending claim 7 recites a step of **defining an object with defined with an option data structure which supports references to option values without preallocation of memory space for the full option values, as recited in parent claim 1, which corresponds to the step of copending claim 1, and notifying objects of a change in an option value through a change handler identified by an option binding, the option binding being located by first searching a mapping data structure for a previously computed mapping to the option binding and, if no mapping was previously computed, by then computing the mapping to the option binding and storing the mapping in the mapping data structure.**

The copending claim 7 does not explicitly recite the additional limitations, as further recited in instant claim 1:

i) **defining an object with defined fields to support values in preallocated memory space** as recited in instant claim 1, but Official Notice is taken that this step was well known in the art of object oriented program development at the time of invention was made: A class definition typically includes one or more fields that declare and store data of various types. The compiler assigns a memory location

(preallocate) to each field in the program based on its type accordingly and then the attribute value of the field is stored in the memory space preallocated to that field -- each instance of a class (object) has its own copy of the fields defined in the class. Therefore, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the step recited in the co-pending claim by adding the step of defining object with defined fields for the purpose of defining and storing data in the memory space preallocated by the compiler. Alternatively, the omission of defined fields from the instant method would be obvious because option values for the purpose of expediting the method.

Ii) **accessing a field value and accessing an option value in the object using expressions of the same syntactic form**, as further recited in instant claim 1. The co-pending claim 7 does not explicitly recite this step. However, both field values and option values are property values of an instance object or a class. Official Notice is taken that this definition was well known in the art of object oriented program development at the time of invention was made: in object oriented programming, properties are usually made as private so that only member methods have access to the properties for purpose of encapsulation. Accessors (getters/setters) are typical of a well-designed object used for accessing a property of a class/object. Regardless of the different memory allocation methods of option values and field values, the use of the same syntax would be obvious because only member methods can access both option and field values. For example, the member method call syntax is objectName.methodName() in Java or C++. The dot operator is used between

class/object name and method name: objectName.getValue(); objectName.setValue(field, option); objectName.fieldValue; objectName.optionValue. It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the step recited in the co-pending claim 7 by adding the step of **accessing a field value and accessing an option value in the object using expressions of the same syntactic form,** as recited in instant claim 1 because both option and field values are properties of the object.

Per claim 17:

The rejection of claims 7 is respectively incorporated, and further, the instant claim recites a system corresponding to copending claim 10 and 19, respectively, modified in the manner set forth above in connection with claim 7 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner set forth above with a system including means for performing the steps of the copending method.

Per claim 23:

The rejection of claims 7 and 17 is respectively incorporated, and further, the instant claim recites a computer product corresponding to copending claim 20, respectively, modified in the manner set forth above in connection with claims 7 and 17 respectively. It would have been obvious for one of ordinary skill in the art of program development to implement the copending method modified in the manner

set forth above with a computer product including means for performing the steps of the copending method. The instant claim recites instructions to notify objects of a change in an option value which corresponds to the step of copending claims 20, except that the instant claim does not explicitly recite how to notify objects. It would have been obvious for one of ordinary skill in the art of program development at the time the instant invention was made to modify the copending method by omitting the method of notifying objects in copending claim 20 for the purpose of expediting the method.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Oath/Declaration

6. The oath or declaration is defective. A new oath or declaration in compliance with 37 CFR 1.67(a) identifying this application by application number and filing date is required. See MPEP §§ 602.01 and 602.02.

> The oath or declaration is defective because: the signature date of oath/declaration for the inventor (Stephen Ward) is missing. A copy of oath/declaration is included in this action.

Information Disclosure Statement

7. The information disclosure statements filed in Paper No. 2, 3, 5, and 6 have been considered.

Specification

8. Applicant is reminded of the proper language and format for an abstract of the disclosure.

The abstract should be in narrative form and generally limited to a single paragraph on a separate sheet within the range of 50 to 150 words. It is important that the abstract not exceed 150 words in length since the space provided for the abstract on the computer tape used by the printer is limited. The form and legal phraseology often used in patent claims, such as "means" and "said," should be avoided. The abstract should describe the disclosure sufficiently to assist readers in deciding whether there is a need for consulting the full patent text for details.

The language should be clear and concise and should not repeat information given in the title. It should avoid using phrases which can be implied, such as, "The disclosure concerns," "The disclosure defined by this invention," "The disclosure describes," etc.

9. The first sentence of abstract is not in narrative form. Also, The abstract exceeds 150 words in length. Appropriate correction is required.

Claim Rejections - 35 USC § 101

10. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claim 25 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claim 25 merely claims as a "computer data signal" that is mere arrangements or compilations of facts, information, or data *per se* and which is merely stored so as to be called "computer-readable" or even outputted by a computer without creating any functional interrelationship, either as part of the stored data or as part of the computing processes performed by the computer ("acts"), then such descriptive material alone

does not impart functionality either to the data as so structured, or to the computer. Thus, such "descriptive material," non-functional descriptive material, that cannot exhibit any functional interrelationship with the way in which computing processes are performed does not constitute a statutory process, machine, manufacture or composition of matter. Also, the purely non-functional descriptive material cannot alone provide the practical application for the manufacture. See In re Warmerdam, 33 F.3d at 1361, 31 USPQ2d at 1760. In re Sarkar, 588 F.2d 1330, 1333, 200 USPQ 132, 137 (CCPA 1978). See Examination Guidelines for Computer-Related Inventions-Final Version, pages 9&10. See MPEP § 2106(IV)(B)(1)(b).

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. Claim 1 is rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant's Admitted Prior Art (APA) disclosed in the instant application.

As per claim 1, APA discloses a method of processing data comprising: defining an object with defined fields to support values in preallocated memory space ("In typical data processing systems,...the compiler creates a data structure. In an object oriented system, the compiler may create a class and a mechanism for creating instances of that

class with defined fields of preallocated memory space into which the values are stored," page 1, lines 8-13) and **with an option data structure which supports references to option values without preallocation of memory space for the full option values (**See APA, page 1 lines 14-19, "An alternative data structure which has, for example, been supported in the [incr Tk] language allows values to be stored in strings or arrays as options associated with an instance object. Using that data structure, memory space is only used for those properties which are given values since the space for those values is not preallocated. Rather, the space is only allocated when a value is optionally added to a list of values associated with the instance object.")

APA does not explicitly disclose the step of **accessing a field value and accessing an option value in the object using expressions of the same syntactic form**, as further recited in instant claim 1. However, Official Notice is taken that this step was well known in the art of object oriented program development at the time of invention was made: both field values and option values are property values of an instance object or a class. In object oriented programming, properties are usually made as private so that only member methods have access to the properties for purpose of encapsulation. Accessors (getters/setters) are typical of a well-designed object used for accessing a property of a class/object. Regardless of the different memory allocation methods of option values and field values, the use of the same syntax would be obvious because only member methods can access both option and field values. For example, the member method call syntax is objectName.methodName() in Java or C++. The dot operator is used between class/object name and method name:

objectName.getValue(); objectName.setValue(field, option); objectName.fieldValue; objectName.optionValue.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include the step of accessing a field value and accessing an option value in the object using expressions of the same syntactic form in the data processing systems disclosed in APA. The modification would be obvious because both option and field values are properties of the object and, therefore, accessed only by member methods to ensure encapsulation.

As per claims 11 and 21, they are system versions of claim 1 and are rejected for the same reasons set forth in connection with the rejection of claim 1 above.

As per claim 22, it is a product version of claim 1 and is rejected for the same reasons set forth in connection with the rejection of claim 1 above.

As per claim 25, it is a computer data signal version of claim 1 and is rejected for the same reasons set forth in connection with the rejection of claim 1 above.

13. Claims 2-3, 5, 6, 8, 11-13, 15, 16, 18, 21, 22, 24, and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant's Admitted Prior Art (APA) disclosed in the instant application in view of "Java: How to Program" by Deitel et al. published in 1997 (hereinafter referred to as "Deitel").

As per claim 2, the rejection of claim 1 is incorporated.

APA does not disclose the use of change handler code. However, Deitel further teaches the use of change handling code in the second data structure to act upon changing option values (See line 67 on page 937, "public void actionPerformed (ActionEvent e)").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use the [incr Tk] option data structure disclosed in the instant application with Dietel's change handling code. One of ordinary skill in the art would have been motivated to process changes to option values.

As per claim 3, the rejection of claim 2 is incorporated.

APA does not disclose the use of a change handler class inheritance hierarchy.

However, in an analogous environment, Deitel further teaches wherein change handler code for one option is defined in different classes within a class inheritance hierarchy and the change handler code from each. Class is executed when the option value changes. See page 347 for a discussion of object-oriented inheritance, lines 25 and 37 on page 526 for an example of change handling code, and the last paragraph on page 527 for a discussion of change handling classes.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use [incr Tk] option data structure disclosed in APA with Dietel's object-oriented inheritance change handling code. One of ordinary skill would have been motivated to use inheritance as a means for code reuse and to ensure a consistent change handling hierarchy.

As per claim 5, the rejection of claim 1 is incorporated. APA does not disclose a default value, the method further comprising, in a get operation to an instance of the class, if an option value which applies to the instance has been set, getting the set option value and, if a value which applies has not been set, getting the default value for the class.

However, in an analogous environment, Deitel teaches the use of "set" and "get" methods. These are used in conjunction with a class called "Time3" which uses and object-oriented constructor to set default values. The get method returns whatever value was last "set" in the variable, including the default value. See pages 308-310 for a discussion of get and set methods, and a definition of the "Time3" class.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use the [incr Tk] option data structure disclosed in the instant application with Dietel's default values and "get" and "set" methods. One of ordinary skill would have been motivated to provide programmers with access to private data members, while retaining the principles of encapsulation.

As per claim 6, the instant application discloses: a first option data structure which supports references to option values without preallocation of memory space for the full option values (See APA, page 1 lines 14-19, "An alternative data structure which has, for example, been supported in the [incr Tk] language allows values to be stored in strings or arrays as options associated with an instance object. Using that data structure, memory space is only used for those properties which are given values since the space for those values is not preallocated. Rather, the space is only allocated when

a value is optionally added to a list of values associated with the instance object."); [incr Tk] is a library of base classes built on [incr Tel] which is an object-oriented extension of the Tel/Tk programming environment.

However, APA does not expressly disclose: defining a first class; defining a second class, the second class data structure form being different from the first class data structure form; and during compilation, encoding an option operation on an option value as a method call to an object of the first class or of the second class without regard to the form of the option data structures supported by the classes.

In an analogous environment, Dietel teaches the use of object-oriented technology including the concepts of encapsulation, inheritance, and polymorphism. Dietel teaches the definition of a first and second class, the second class data structure form being different from the first class data structure form. See page 935 for a definition of class "VectorTest", and page 950 for a definition of the class "PropertiesTest". Class VectorTest uses a "Vector" data structure form (see line 20 on page 936: "v = new Vector (1)"), while class PropertiesTest uses a "Properties" data structure form (see line 22 on page 951: "table = new Properties();"), which is based on a hash table.

Also, Dietel teaches compilation without regard to the form of the data structures supported by the classes through the use of dynamic method binding (See page 370, Section 7.12, "Dynamic Method Binding;" "Suppose a set of shape classes such as Circle, Triangle, Rectangle, Square, etc. are all derived from superclass Shape. In object-oriented programming, each of these classes might be endowed with the ability

to draw itself... Then to draw any shape, we could simply call method draw of superclass Shape and let the program determine dynamically (i.e. at execution time) which subclass draw method to use"). Implicitly, the compiler does not differentiate the form of the option data structures supported by the class, since this is handled at execution time (see page 382, Section 7.17, "New Classes and Dynamic Binding": "New classes are accommodated by dynamic method binding (also called late binding).

An object's type needs not be known at compile time for a polymorphic call to be compiled. At execution time, the call is matched with the method of the called object"). Encoding an operation on a value as a method call to an object is implicit in the operation of compilers for object-oriented programming languages.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use the [incr Tk] option data structure disclosed in the instant application with Dietel's object-oriented programming techniques. One of ordinary skill would have been motivated to include object-oriented techniques to promote reuse, data encapsulation, and class inheritance as object-oriented technology allows dynamic binding of method calls and multiple implementations of the same method.

As per claim 8, the rejection of claim 1 is incorporated. APA does not expressly disclose the use of a linked list of option items.

However, in an analogous environment, Deitel teaches the use of linked lists as a way to store option items of any type, including objects of other classes. See page 884-894 for a discussion and implementation of a linked list data structure.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use [incr Tk] option data structure disclosed in APA with Dietel's linked list. One of ordinary skill would have been motivated to use a linked list since the number of data elements to be represented in the data structure at one time is unpredictable.

As per claims 12-13, 15, 16, and 18, they are system versions of claims 2-3, 5, 6, and 8 and are rejected for the same reasons set forth in connection with the rejection of claims 1-3, 5, 6, and 8 above.

As per claim 24, see the rejection of claim 8 above.

14. Claims 7, 17 and 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant's Admitted Prior Art (APA) in the background section of the instant application in view of Deitel ("Java: How to Program" published in 1997) as applied to claims 7, 17 and 23 , and further in view of "Curl: A Gentle Slope Language for the Web" by Hostetter et al. published in 1997 (hereinafter referred to as Hostetter) and further in view of "Advanced Compiler Design & Implementation" by Muchnick pulished in 1997 (hereinafter referred to as Muchnick).

As per claim 7, Deitel teaches the use of method binding, as discussed in the rejection of claim 1, to bind methods and options (See page 370). APA does not

expressly disclose notifying objects of a change in an option value through a change handler identified by an option binding, the option binding being located by first searching a mapping data structure for a previously computed mapping to the option binding and, if no mapping was previously computed, by then computing the mapping to the option binding and storing the mapping in the mapping data structure.

However, in an analogous environment, Hostetter teaches the use of a dynamically bound environment implemented using a deep binding mechanism. Notification of a change in a property, or option, is propagated through a tree of objects that may, in turn, cause various reformatting operations to take place. Reformatting operations are initiated by a change handler and controlled by the tree. A mapping data structure is inherently used to search the tree. Hostetter teaches that an upward search of the template/instance tree is performed starting with the current object.

Muchnick further teaches the storage of option bindings in a symbol table using procedures such as "Set Sym-Attr" which sets the given attribute of the given symbol in the given symbol table to the given value.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the programming language [incr Tk] disclosed in APA with Deitel's object-oriented programming language and dynamic method binding, along with Hostetter's search tree and Muchnick's symbol table. One of ordinary skill would have been motivated to use a tree for efficient searching. One of ordinary skill would have also been motivated to use a function to set attributes such as option bindings in order to manage data and reduce search time.

As per claim 17, it is a system version of claim 7, and is rejected for the same reasons set forth in connection with the rejection of claim 7 above.

As per claim 23, see the rejection of claim 7 above.

15. Claims 4, 9, 10, 14, 19, and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant's Admitted Prior Art (APA) disclosed in the background section of the instant application, in view of Deitel as applied to claim 1 above, and further in view of "Curl: A Gentle Slope Language for the web" by Hostetter et al. published in 1997 (hereinafter referred to as Hostetter).

As per claim 9, APA does not expressly disclose the use of nonlocal option values that apply to other objects in a nonlocal option hierarchy.

However, in an analogous environment, Hostetter teaches the use of nonlocal option values, or properties, which apply to other objects in a nonlocal option hierarchy, or tree. See the top third of page 10.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use the [incr Tk] option data structure disclosed in APA with the Hostetter's nonlocal option values. One of ordinary skill would have been motivated to propagate option values through a hierarchy.

As per claim 10, the rejection of claim 9 is incorporated.

APA does not expressly disclose the use of a graphical hierarchy.

However, in an analogous environment, Hostetter teaches using nonlocal properties with graphic objects in a hierarchical tree. See the top third of page 10.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use the [incr Tk] option data structure disclosed in APA with the Hostetter's graphical hierarchy. One of ordinary skill would have been motivated use a graphical hierarchy to re-render objects.

As per claims 19 and 20, they are system versions of claims 9 and 10, and are rejected for the same reasons set forth in connection with the rejection of claims 9 and 10 above.

As per claim 4, APA does not explicitly disclose the teaching for using the type description in the option data structure to process an operation on the option value during compilation.

Hostetter, however, discloses the method further comprising: during compilation, using the type description in the option data structure to process an operation on the option value (Page 1, Lines 22-29," Programming complex operations. Other components of an interactive document may require more sophisticated mechanisms than are provided by the interface toolkit. These components can also be developed using Curl since, at its heart, Curl is really an object-oriented programming language. Curl expressions, class definitions and procedure definitions embedded in the web document are securely compiled to native code by the built-in on-the-fly compiler and then executed without the need for any sort of interpreter. Curl provides many of the features of a modern object-oriented programming language: multiple inheritance, extensible syntax, a strong type system that includes a dynamic "any" type, safe execution through encapsulation of user code and extensive checking performed both

at compile and run time. Almost all of the Curl system and compiler are written in Curl;" page 7, lines 11-12, "Lexically-scoped environment. Curl provides a structured name space whose bindings include variables, constants, types, and compilation hooks for arbitrary syntactic forms.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use the [incr Tk] option data structure disclosed in APA with the Hostetter's method. The modification would have been obvious because one of ordinary skill in the art would have been motivated to use Curl modern object-oriented programming feature (object structure) to compile to the native code and "execute without the need for any sort of interpreter."

As per claim 14, see the rejection of claim 4 above.

Claim Rejections - 35 USC § 102

16. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

17. Claims 1-3, 7, 11-13, 17, 21-23, and 25 are rejected under 35 U.S.C. 102(b) as being anticipated by Berry et al. (U.S. Patent No. 5,732,271, hereinafter "Berry").

As per claim 1, Berry discloses a method and system which provides a prototypical object (object /data structure) that can be copied to create a derived object (See abstract) for processing data comprising: defining an object with defined fields to

support values in preallocated memory space (Column 2, Lines 37-41, "The foregoing objects are achieved by a method and system which provides a prototypical object which can be copied to create a derived object. A derived object can contain attribute values or it can hold references to prototypical objects;" Column 3, Lines 51-54, "For example, button object 220 can contain attributes such as background color, button action and size. Rectangle object 214, on the other hand, can contain attributes such as fill color, contents, border type and size.") and with an option data structure which supports references to option values without preallocation of memory space for the full option values (see Column 2, Lines 37-41, "The foregoing objects are achieved by a method and system which provides a prototypical object which can be copied to create a derived object. A derived object can contain attribute values or it can hold references to prototypical objects;" Column 3, Lines 51-54, "For example, button object 220 can contain attributes such as background color, button action and size. Rectangle object 214, on the other hand, can contain attributes such as fill color, contents, border type and size;" The Examiner interprets only attribute values contained in the derived object need memory space preallocated; for those reference it holds do not require memory preallocation (already allocated by the prototypical objects)); and accessing a field value and accessing an option value in the object using expressions of the same syntactic form (Column 2, Lines 37-41, "The foregoing objects are achieved by a method and system which provides a prototypical object which can be copied to create a derived object. A derived object can contain attribute values or it can hold references to prototypical objects;" Column 3, Lines 51-54, "For example, button object 220 can

contain attributes such as background color, button action and size. Rectangle object 214, on the other hand, can contain attributes such as fill color, contents, border type and size," see also col 3, lines 39-64; see also the double patenting rejection of claim 1).

As per claim 2, Berry discloses a method as claimed in claim 1 wherein the option data structure identifies change handler code that is executed when an option value changes (Column 2, Lines 41-50, "If a required value is not held by a prototypical object, the present invention discloses a scheme by which the object searches up an object hierarchy to find the required attribute. In addition, each object can register interests in prototypical objects. If an attribute of a prototypical object changes, the prototypical object informs all registered objects of the change. At runtime, the prototypical object becomes a master object whose attribute values can be changed by the user. Changes in master object attributes are propagated to all registered derived objects.").

As per claim 3, the rejection of claim 2 is incorporated and Berry further discloses a method as claimed in claim 2 wherein change handler code for one option is defined in different classes within a class inheritance hierarchy and the change handler code from each class is executed when the option value changes (see Column 2, Lines 41-50, "If a required value is not held by a prototypical object, the present invention discloses a scheme by which the object searches up an object hierarchy to find the required attribute. In addition, each object can register interests in prototypical objects. If

an attribute of a prototypical object changes, the prototypical object informs all registered objects of the change. At runtime, the prototypical object becomes a master object whose attribute values can be changed by the user. Changes in master object attributes are propagated to all registered derived objects;" Column 4, Lines 22-31, "FIG. 3 depicts top card 310 of the attribute sheet for button object 220. Button 220 has one attribute: background color. This attribute is indicated by tab 312 associated with the top card 310. To change the background color for button object 220 the developer selects tab 312 on attribute sheet 310, then the developers selects a color from the displayed palette of rectangle objects 316. After a color is selected, the developer can edit the color by pressing button 318, select the color by closing the attribute sheet using window button 324 or cancel the selection by pressing button 320.").

As per claim 7, Berry discloses a method as claimed in claim 1 further comprising: notifying objects of a change in an option value through a change handler identified by an option binding, the option binding being located by first searching a mapping data structure for a previously computed mapping to the option binding and, if no mapping was previously computed, by then computing the mapping to the option binding and storing the mapping in the mapping data structure (see Column 2, Lines 41-50, "If a required value is not held by a prototypical object, the present invention discloses a scheme by which the object searches up an object hierarchy to find the required attribute. In addition, each object can register interests in prototypical objects. If an attribute of a prototypical object changes, the prototypical object informs all registered objects of the change. At runtime, the prototypical object becomes a master

object whose attribute values can be changed by the user. Changes in master object attributes are propagated to all registered derived objects.").

As Per Claim 17, the rejection of claim 11 is incorporated respectively and further Berry discloses change handlers which notify objects of a change in an option value and a mapping data structure which maps an option name and class to an option binding which identifies a change handler (Column 2, Lines 41-50, "If a required value is not held by a prototypical object, the present invention discloses a scheme by which the object searches up an object hierarchy to find the required attribute. In addition, each object can register interests in prototypical objects. If an attribute of a prototypical object changes, the prototypical object informs all registered objects of the change. At runtime, the prototypical object becomes a master object whose attribute values can be changed by the user. Changes in master object attributes are propagated to all registered derived objects.").

As per claims 11-13 and 21, they are the system versions of claims 1-3, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 1-3 above.

As per claims 22-23, they are the product versions of claims 1 and 7, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 1 and 7 above.

As per claim 25, it is the data signal version of claims 1, 11, 21, and 22, respectively, and is rejected for the same reasons set forth in connection with the rejection of claims 1, 11, 21, and 22 above.

Claim Rejections - 35 USC § 103

18. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 4-6, 8-10, 14-16, 18-20, and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Berry et al. (US Patent No. 5,732,271, art of record, hereinafter "Berry"), in view of Hostetter et al., "Curl: A Gentle Slope Language for the Web," World Wide Web Journal, spring, 1997, art of record, hereinafter "Hostetter").

As per claim 4, Berry discloses a method of processing data comprising defining an object with defined fields and with an option data structure accessing a field value and an option value using the same syntactic form. Berry, however, does not explicitly disclose his teaching for using the type description in the option data structure to process an operation on the option value during compilation.

Hostetter discloses the method further comprising: during compilation, using the type description in the option data structure to process an operation on the option value (Page 1, Lines 22-29," **Programming complex operations.** Other components of an interactive document may require more sophisticated mechanisms than are provided by the interface toolkit. These components can also be developed using Curl since, at its heart, Curl is really an object-oriented programming language. **Curl expressions, class definitions and procedure definitions embedded in the web document are securely compiled to native code by the built-in on-the-fly compiler and then**

executed without the need for any sort of interpreter. Curl provides many of the features of a modern object-oriented programming language: multiple inheritance, extensible syntax, a strong type system that includes a dynamic "any" type, safe execution through encapsulation of user code and extensive checking performed both at compile and run time. Almost all of the Curl system and compiler are written in Curl;" page 7, lines 11-12, **"Lexically-scoped environment.** Curl provides a structured name space whose bindings include variables, constants, **types, and compilation** hooks for arbitrary syntactic forms.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry. The modification would have been obvious because one of ordinary skill in the art would have been motivated to use Curl modern object-oriented programming feature (object structure) to compile to the native code and "execute without the need for any sort of interpreter."

As per claim 5, Berry does not explicitly disclose getting the option value. However, Hostetter teaches an option data structure includes a default value, the method further comprising, in a get operation to an instance of the class, if an option value which applies to the instance has been set, getting the set option value and, if a value which applies has not been set, getting the default value for the class (see Section3, Page 4, Lines 1-2, "The screen shot above reflects the fact the user has selected something **besides the default** color (red) and quantity (0).").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry, to get the option value. The modification would have been obvious because one of ordinary skill in the art would have been motivated to incorporate a dynamic object with a simple mechanism for propagating changes in its value to other dynamic objects that depend on first object's value and to customize the object using the option value.

As per claim 6, Berry does not explicitly disclose encoding an option operation. However, Hostetter teaches defining a first class with a first option data structure of a first form which supports, in instances of the class, references to option values without preallocation of memory space for the full option values (see Page 4, Figure 2, item hbox); defining a second class with a second option data structure of a second form which supports, in instances of the second class, references to option values without preallocation of memory space for the full option values, the second form being different from the first form (see Page 4, Figure 2, item vbox) and during compilation, encoding an option operation as a method call to an object of the first class and to an object of the second class without regard to the form of the option data structure supported by the class (see Page 4, Figure 2, item hbox and item vbox; Page 3, Line 20-24, "Since the values for color and quantity are Dynamic objects, the last line of the display changes automatically as the user manipulates the color and quantity controls. A Dynamic object incorporates a simple mechanism for propagating changes in its value to other dynamic objects that depend on first object's value. More sophisticated

propagation rules could be supplied by the user by creating a new class of objects derived from Dynamic objects that have a different "propagate" method;" Page 9, Lines 20-22, "Hboxes and vboxes. These are one-dimensional formatters that create simple horizontal or vertical arrangements of their children, lining up their baselines or margins. As in TeX, the relative allocation of white space is controlled by the elasticity of any glue objects that have been added as children").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry. The modification would have been obvious because one of ordinary skill in the art would have been motivated to display changes automatically as the user manipulates the color and quantity controls.

As per claim 8, Berry does not explicitly disclose an option list. However, Hostetter teaches that the option data structure comprises a linked list of option items having option values (see Section 4.3, Page 10, Lines 21-22, "Much of the flexibility of boxes comes from the use of properties to control the rendering of primitive objects. A property is a (name, value) binding and each Graphic object has an associated list of properties."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry. The modification would have been obvious because one of ordinary skill in the art would have been motivated to implement properties in a dynamically bound environment using a deep binding mechanism and to facilitate the selection of options using option list.

As per claim 9, Berry does not explicitly disclose a nonlocal option value applies to other objects in a nonlocal option hierarchy. However, Hostetter teaches a nonlocal option value applies to other objects in a nonlocal option hierarchy (see Section3, Page 4, Lines 1-2, "The screen shot above reflects the fact the user has selected something besides the default **color** (red) and quantity (0)."). Color is a nonlocal option because all text in a given document is usually the same color.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry, to comprise a nonlocal option value that applies to other objects in a nonlocal option hierarchy. The modification would have been obvious because one of ordinary skill in the art would have been motivated to implement properties in a dynamically bound environment using a deep binding mechanism.

As per claim 10, Berry does not explicitly disclose that the nonlocal option hierarchy is a graphical hierarchy. However, Hostetter teaches that the nonlocal option hierarchy is a graphical hierarchy *(see* Section3, Page 4, Lines 12, "The screen shot above reflects the fact the user has selected something besides the default color (red) and quantity (0);" Section 4.3, Page 9, Lines 34-35, "text. Properties control the color, size and font family as well as indicating whether the text should be bold or italic.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry. The modification would have been obvious because one of ordinary skill in the art would have been motivated to represent a graphic image as a hierarchical tree of

Graphic objects (Leaves of the tree are primitive Graphic objects which know how to draw themselves, usually after looking up the values of various properties).

As per claim 16, Berry does not explicitly disclose plural classes having data structures of different forms. However, Hostetter teaches plural classes having data structures of different forms, and a compiler which encodes an option operation as a method call to an instance object of one of the classes without regard to the form of the option data structure supported by the class. (See Page 4, Figure 2, item hbox and item vbox; Page 4, Figure 2, item hbox and item vbox; Page 3, Line 20-24, "Since the values for color and quantity are Dynamic objects, the last line of the display changes automatically as the user manipulates the color and quantity controls. A Dynamic object incorporates a simple mechanism for propagating changes in its value to other dynamic objects that depend on first object's value. More sophisticated propagation rules could be supplied by the user by creating a new class of objects derived from Dynamic objects that have a different "propagate" method;" page 9, Lines 20-22, "Hboxes and vboxes. These are one dimensional formatters that create simple horizontal or vertical arrangements of their children, lining up their baselines or margins. As in TeX, the relative allocation of white space is controlled by the elasticity of any glue objects that have been added as children.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hostetter into the method of Berry, to comprise plural classes having data structures of different forms. The modification would have been obvious because one of ordinary skill in the art would

have been motivated to display changes automatically as the user manipulates the color and quantity controls.

As per claims 14, 15, and 18-20, they are the system versions of claims 4, 5, and 8-10, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 4, 5, and 8-10 above.

As per claim 24, it is the product version of claims 8 and 18, respectively, and is rejected for the same reasons set forth in connection with the rejection of claims 8 and 18 above.

19. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Insun Kang whose telephone number is 703-305-6465. The examiner can normally be reached on M-F 8:30-5:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on 703-305-9662. The fax phone number for the organization where this application or proceeding is assigned is 703-308-3988.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 703-305-3900

IK

12/9/2004

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100